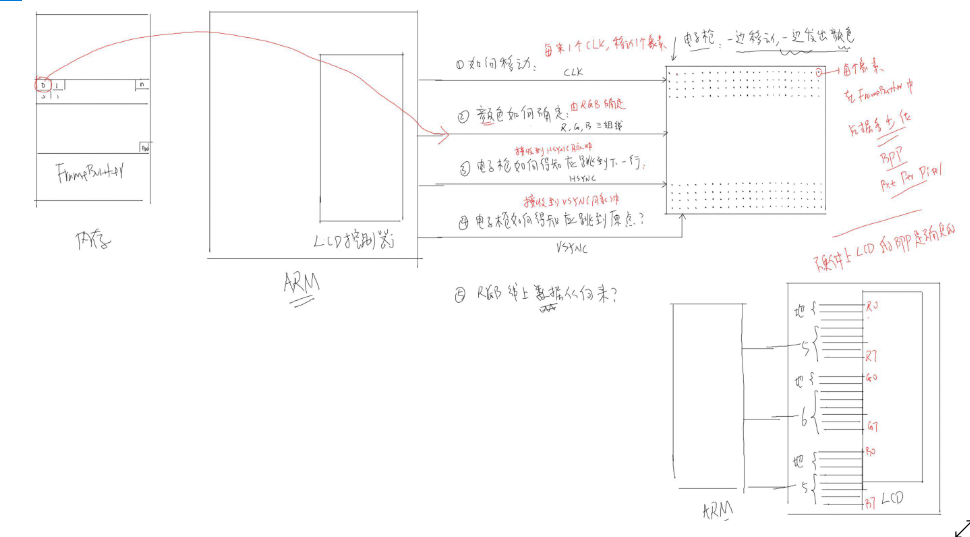
第017课 LCD

第001节\_LCD硬件原理



LCD控制器控制发出这些控制信号：

1.如何移动像素点

每次发一个clk,电子枪移动一个像素点。

2.确定颜色

RGB 三组线

3.跳到下一行？

HSYNC（水平同步信号）

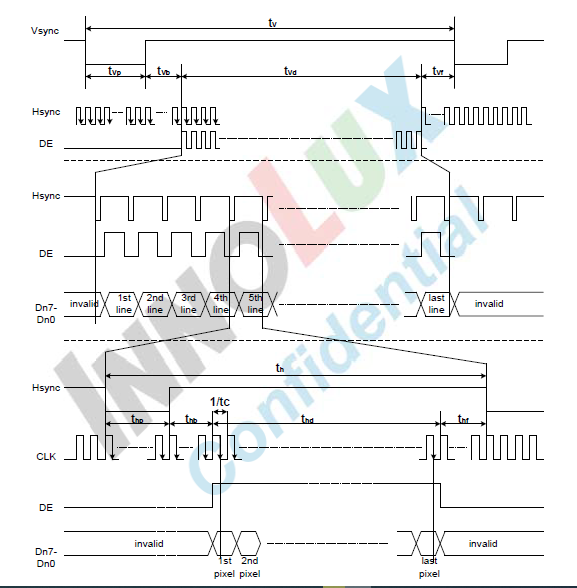
4.电子枪如何跳到原点。

VSYNC(垂直同步信号)

在内存中会划分区域给freebuffer,他会存贮像素点的颜色的值，lcd控制器会循环会通过RGB 三组线传给电子枪，电子枪将其转化为红绿蓝打到屏幕上。

脉冲宽度

移动时间



显示玩最右像素再过多少时间hsclk才到

显示完最下一行再过多久vsyanc 才到

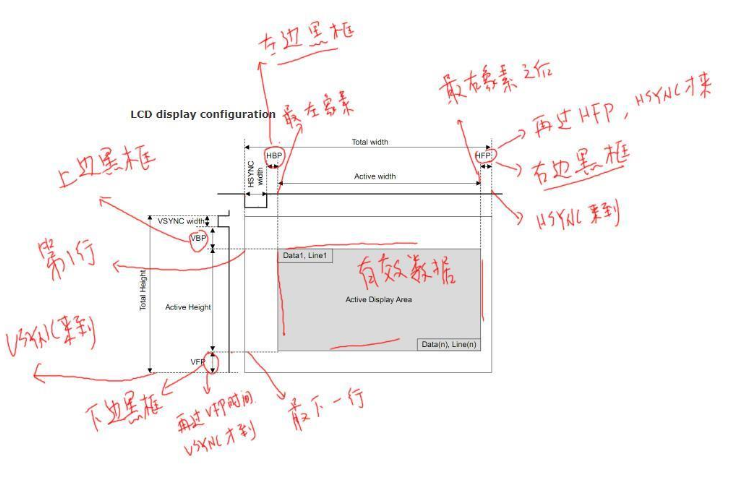
多少行

脉冲宽度

移动时间

Freebuffe

Lcd控制器在电子枪的clk下降沿将从数据线得到数据。发送电子枪 电子枪转化为颜色打到像素点上。



中间是有效数据。RGB 每个8位。所以一个像素点需要24条线

但是2440只给他16条线。

参考文章:

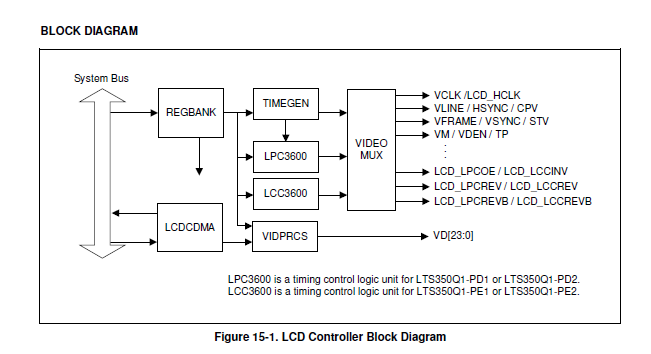
http://www.cnblogs.com/shangdawei/p/4760933.html

第002节\_S3C2440\_LCD控制器

功能:

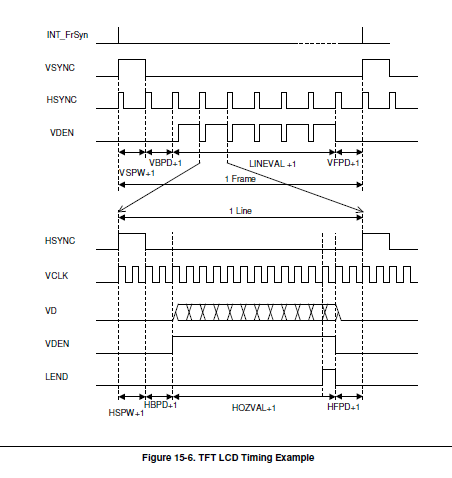
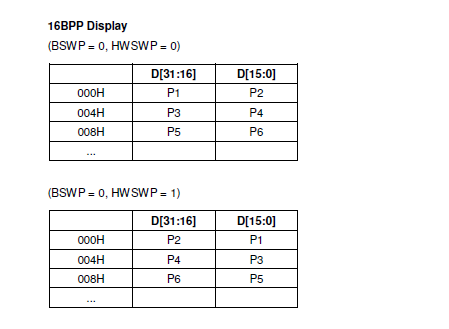
a. 取数据 : 把framebuffer的地址告诉LCD控制器, bpp（bit per pixel 像素的位数）, 分辨率

b. 发数据 : 把时序告诉LCD控制器、设置引脚的极性



内存

从内存的freebuffer 取数据发给LCD (LCDCDMA 直接取数据不需要cpu)



根据外接的lcd设置

假设 arm板 与lcd 之间16个数据线，在FB中每个像素点是16bit的数据。

但是要考虑节省空间用8bit表示。所以引入调色板（特殊的内存）。

FB 8bit（伪彩色）数据 发给lcd控制器 –lcd控制器从调色板中拿出真正的16bit数据。

调色版的的地址占据四字节其实只用到两字节。

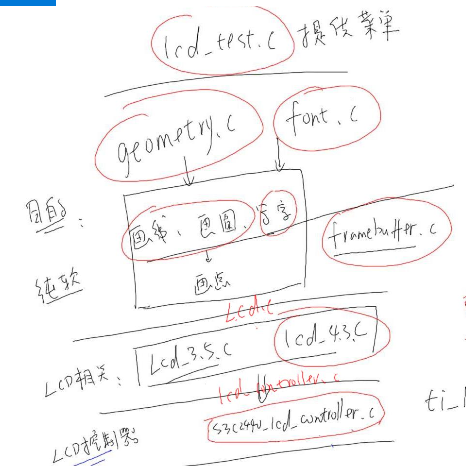
如果要将lcd设置为同一种颜色，16bit 或者 24bit 只能将fb设置为同一种值。但对于8bit 调色板或fb两者都行。还有一种临时调色板（直接从临时调色板拿不从fB）

第003节\_编程\_框架与准备

a. 讲解后续程序的框架

b. 准备一个支持NAND、NOR启动的程序

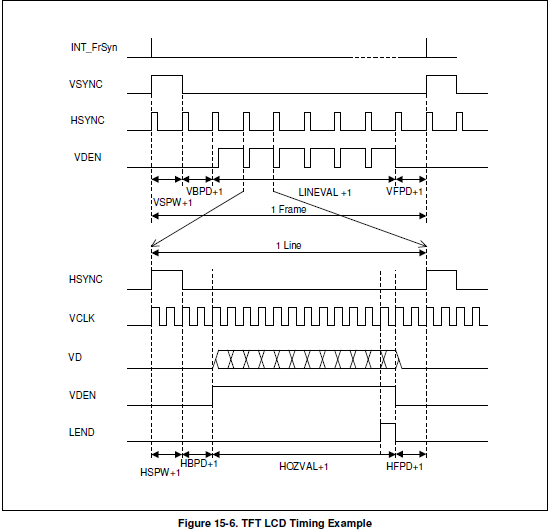
大致的框架如下：



面向对象的机构化编程：

第004节\_编程\_抽象出重要结构体

第005节\_编程\_LCD控制器



VBPD+1 = tvb（移动时间）注释：参考lcd图

LINEVAL = line-1

VFPD = tvf -1

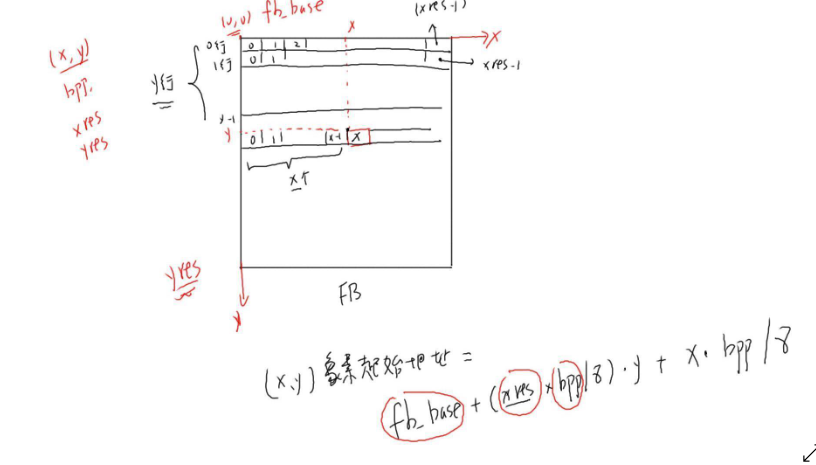
VSPW = tvp-1(tvp 脉冲宽度)

第006节\_编程\_LCD设置

第007节\_编程\_简单测试

第008节\_编程\_画点线圆

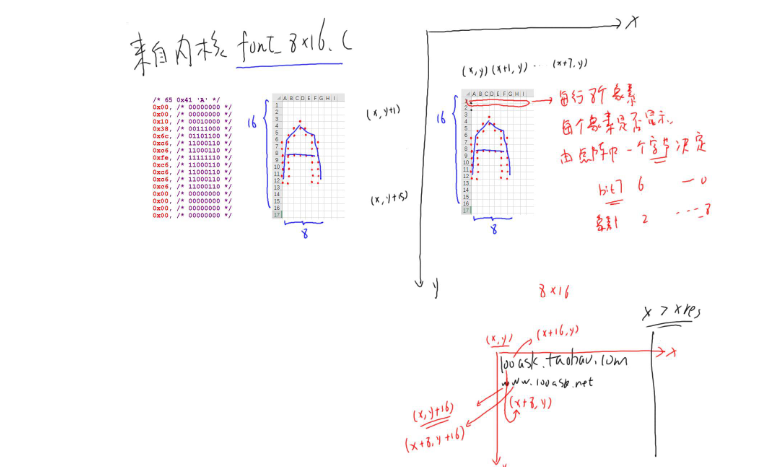
要画一个点：要知道fb的基地址以及x y的坐标



根据上图可以推算出x y的坐标为：FB\_base + (xres \*bpp/8)\*(y-1) + x \*bpp/8

百度搜索：c语言lcd画圆

第009节\_编程\_显示文字



从内核中找到front.c.根据A的子模。可以描绘出Ａ

第010节\_编程\_添加除法

因为在做除法计算clkval的时候报错

undefined reference to `\_\_floatsidf'

对于未实现的函数:

a. 去uboot中查找

b. 去内核源码中查找

c. 去库函数中查找 (一般来说编译器自带有很多库)

arm-linux-gcc -v

cd /work/tools/gcc-3.4.5-glibc-2.3.6 进入bin的上一层

进入工具链的目录: grep "\_\_floatsisf" \* -nR 找到.a文件

如果换了编译器需求要区编译器的目录找到相应的lib.a需要一个个自己去尝试

第011节\_编程\_使用调色板

作业：

1. 如果有MINI2440、TQ2440或是带3.5寸LCD的JZ2440，添加一个lcd\_xxx.c文件，构造lcd\_params结构体，

体验结构化编程的优点：可以很轻松支持其他LCD

2. 找到汉字库点阵，在LCD上显示汉字

这需要你善用百度，找到汉字库，也许要阅读别人的代码了解汉字库点阵的存储方式。

这是一个综合能力的体现。